Stratégies d'observation dans les simulations orientées agent

G. Morvan[a] A. Veremme[a,b] D. Dupont[b] D. Jolly[a]
gildas.morvan alexandre.veremme daniel.dupont daniel.jolly

[a]Univ Lille Nord de France, F-59000 Lille France,
U. Artois, LGI2A EA 3926 Technoparc Futura, F-62400 Béthune, France
@fsa.univ-artois.fr

[b]Département Ingénierie et Sciences du Vivant,
HEI, 13 rue de Toul, F-59046 Lille Cedex, France
@hei.fr

Résumé

L'observation de simulations orientées agent composées de nombreux agents est une tâche coûteuse en temps de calcul. Dans cet article, nous proposons deux méthodes, très simples à mettre en œuvre, basées sur la théorie des sondages et la notion d'auto-observation, permettant d'optimiser le calcul d'observables. Ces méthodes sont évaluées et comparées empiriquement sur un problème jouet.

Mots-clés : *Simulation multi-agents, observation*

Abstract

The observation of agent-based simulations composed of numerous agents is a task requiring a lot of computation time. In this paper, we introduce two methods, easy to implement, based on survey theory and the notion of auto-observation, to optimize the computation of observables. These methods are evaluated and compared empirically on a toy problem.

Keywords: *Agent-based simulation, observation*

1 Introduction

Les avancées à la fois théoriques et pratiques dans le domaine de la simulation orientée agent (SOA) ont permis la modélisation de systèmes toujours plus complexes, *i.e.*, formés de composants en interaction toujours plus nombreux. L'exécution et l'analyse des systèmes artificiels résultants posent de nombreux problèmes d'ingénierie. Ainsi, un problème classique lié aux SOA et plus généralement aux systèmes multi-agents (SMA) est leur observation. Ces systèmes étant composés de nombreuses entités indépendantes, il est souvent nécessaire d'agréger leurs états pour fournir à l'utilisateur un ou plusieurs indicateurs de l'état du système dans sa globalité. On trouve dans la littérature relative au problème de l'observation en temps réel (ou monitoring) d'agents deux familles de méthodes :

– l'observation basée sur les rapports d'agents (report-based monitoring) suppose que chaque agent du système envoie un message contenant les valeurs des propriétés à observer à un « agent superviseur » qui agrégera ces données [11],
– l'observation indirecte consiste à écouter les conversations des agents (ou plus généralement toute conséquence facilement observable de l'activité des agents) pour inférer les valeurs des propriétés à observer [4].

Kamika et al. [4] soulignent que, dans le cadre des SMA, l'observation indirecte est intéressante comparativement à l'obervation directe pour au moins deux raisons :

1. il n'est pas nécessaire de modifier les agents pour la mettre en œuvre,
2. son impact sur le temps de calcul est faible.

En revanche, il n'est pas toujours possible de mettre en œuvre une telle méthode. En effet, rien ne garantit qu'une propriété du système puisse être observée indirectement. Notons cependant que l'observation des interactions, et en particulier des conversations entre agents d'une simulation, présente un intérêt intrinsèque, notamment pour comprendre, débugger ou encore valider une SOA [1]. Des outils basés sur ce principe ont ainsi été développés [9].

Les objectifs de l'observation d'agents dans les SMA (au sens général) et les SOA peuvent être, si l'on se réfère à la littérature, très différents. Cela est dû aux caractéristiques des agents (notamment leur capacité à élaborer collectivement des plans), du médium de communication (*e.g.*, l'accessibilité), du système (physiquement distribué ou non) et des observations elles-mêmes. Ainsi, dans le cas des SOA qui nous occupe ici,

il est possible de distinguer certains de ces buts, liés aux différentes étapes de développement du modèle [10]. Une simulation sera ainsi observée afin de fournir à l'utilisateur ou à un système automatisé d'analyse de simulations des informations sur l'état du système pour :
– vérifier le simulateur,
– valider le modèle,
– valider une simulation ou certains paramètres de la simulation.

Par ailleurs, les méthodes d'observation sont elles aussi différentes. Les méthodes d'observation de SMA citées plus haut, ont été essentiellement conçues pour des systèmes composés d'agents fonctionnellement autonomes (aucun système externe à l'agent ne peut accéder à sa structure ou aux valeurs de ses propriétés). Dans le cas des SOA, il ne semble pas aberrant que des agents extérieurs à la simulation puissent accéder aux valeurs des propriétés des agents de la simulation. En d'autres termes, si les agents de la simulation sont fonctionnellement autonomes entre eux, rien n'impose qu'ils ne le soient des systèmes d'observation et d'exécution de la simulation. Ainsi, les plates-formes de SOA fournissent généralement des outils d'observation fondés sur la notion de « sondage » (probing), *i.e.*, d'accès direct aux propriétés à observer : un agent observateur ayant pour rôle le calcul d'un observable sonde l'ensemble des agents du système pour déterminer les agents devant être observés et agréger les états de ceux-ci.

L'utilisation de ces outils a bien évidemment un coût sur le temps de calcul de la simulation. Leur impact dépend de la complexité de l'algorithme d'agrégation, de la fréquence des observations et, selon la nature de l'observation, du nombre d'agents, de la complexité des organisations et des interactions ou encore de la taille de l'environnement. Il n'existe pas, à notre connaissance, de travail existant relatif à ce problème[1].

Dans cet article, nous introduisons deux méthodes d'observation directe de SOA et analysons de manière comparative leurs performances sur un cas d'étude simple présenté dans la section 2. La première méthode, basée sur le concept d'auto-observation, consiste à intégrer des fonctions d'observation locale dans le cycle de vie des agents de la simulation. La seconde, largement utilisée dans les sciences humaines (*e.g.* le marketing, la sociologie ou encore la démographie), est basée sur la théorie des sondages. Toutes deux sont non-intrusives, dans le sens où la plate-forme n'a pas été modifiée pour les mettre en œuvre. Dans la section 3, nous montrons que ces méthodes peuvent être des alternatives intéressantes au sondage de l'ensemble des agents. Nous nous focalisons sur les simulations composées d'un très grand nombre d'agents, *i.e.*, pour lesquelles le temps de calcul nécessaire à l'observation de la simulation devient grand par rapport à celui de la simulation elle-même.

[1] Les publications relatives à l'observation de SOA, *e.g.*, [7], traitent de la représentation des informations, non de leur calcul.

2 Cas d'étude

Nous considérons des SOA particulièrement simples, inspirées du *StupidModel* [8], afin de mettre en œuvre les méthodes décrites dans les sections suivantes. Ces SOA sont caractérisées par :
– un environnement toroïdal $\mathcal{E}$ de dimension 2 composé de $100 \cdot 100$ cellules carrées (avec le voisinage de Moore),
– N agents se déplaçant aléatoirement dans cet environnement durant 100 pas de temps,
– l'observation du nombre d'agents Z présents dans une zone de l'environnement $\mathcal{Z} \subseteq \mathcal{E}$, définie arbitrairement (surface et position) à chaque pas de simulation.

L'efficacité des différentes méthodes d'observation sera analysée en fonction de N et de $E(Z)$, l'espérance mathématique du nombre d'agents présents dans $\mathcal{Z}$. Nous considérerons ici que

$$E(Z) = N \cdot \frac{\text{surface de } \mathcal{Z}}{\text{surface de } \mathcal{E}} \qquad (1)$$

Ces simulations ont été implémentées sur la plate-forme MadKit/TurtleKit [3, 5]. Chaque simulation est répétée 10 fois. Cependant, la dispersion des temps de calcul s'est révélée très faible. À titre d'exemple, la figure 1 compare les temps CPU nécessaires au calcul de telles simulations, pour $E(Z) = N/2$, en fonction de N, lorsqu'on les observe à l'aide de l'outil fourni dans la plate-forme (la méthode d'observation sur laquelle est basée cet outil, fondée sur l'observation de l'ensemble des agents, sera appelée « méthode naïve » par la suite) et lorsque l'on ne les observe pas.

Dans les sections suivantes, nous proposons des idées pour réduire l'impact des méthodes d'observation directes sur le temps de calcul des simulations.

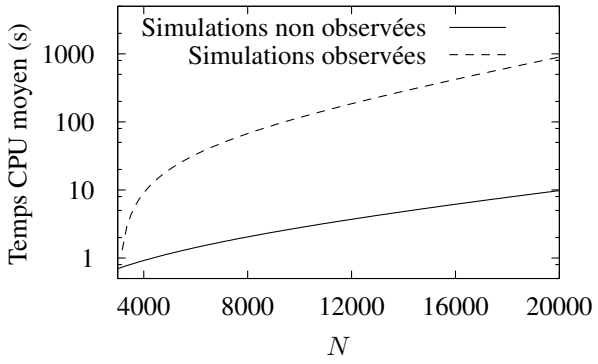

FIG. 1 – Temps CPU moyen utilisé pour le calcul de différentes simulations avec et sans observation pour $E(Z) = N/2$

3 Filtrage des agents observés

Dans le cas d'étude considéré, comme dans de nombreux cas, il n'est pas nécessaire de sonder tous les agents pour calculer la valeur de la propriété observée. Formellement, on considère une fonction d'observation

$$obs : 2^{\mathcal{A}} \rightarrow \mathcal{I}, \tag{2}$$

où $\mathcal{A}$ représente l'ensemble des agents et $\mathcal{I}$ l'ensemble des valeurs pouvant être observées. On s'intéresse au sous-ensemble d'agents minimal $\mathcal{A}' \subseteq \mathcal{A}$ défini par un ensemble de contraintes (*e.g.*, dans l'exemple précédent on considère une contrainte unique relative à la position de l'agent dans l'environnement) permettant de calculer obs correctement. En d'autres termes, on cherche à déterminer $\mathcal{A}'$ tel que

$$\begin{array}{c} obs'(\mathcal{A}') = obs(\mathcal{A}) \text{ et} \\ \nexists\, \mathcal{A}'' \subset \mathcal{A}' \mid obs'(\mathcal{A}'') = obs(\mathcal{A}), \end{array} \tag{3}$$

où obs' est une fonction d'observation « simplifiée », *i.e.*, pouvant être calculée plus rapidement que obs car spécifique à $\mathcal{A}'$:

$$\begin{array}{c} \forall\, \mathcal{A}'' \neq \mathcal{A}' \subseteq \mathcal{A},\ obs'(\mathcal{A}') = obs(\mathcal{A}') \\ \text{mais } obs'(\mathcal{A}'') \neq obs(\mathcal{A}''). \end{array} \tag{4}$$

Ainsi, dans notre exemple, la fonction d'observation $obs(\mathcal{A})$ filtre les agents de $\mathcal{A}$ afin de ne compter que ceux situés dans $\mathcal{Z}$, alors que $obs'(\mathcal{A}')$ ne fait que compter les agents de $\mathcal{A}'$.

Afin de pouvoir utiliser obs', il est nécessaire de considérer une fonction de filtrage capable d'identifier le sous-ensemble $\mathcal{A}'$ (dans notre exemple ce sous-ensemble ne contient bien évidemment que les agents situés dans $\mathcal{Z}$) :

$$\begin{array}{c} filtre : 2^{\mathcal{A}} \rightarrow 2^{\mathcal{A}} \mid \forall \mathcal{A}', \mathcal{A}'' \subseteq \mathcal{A}, \\ \text{si } filtre(\mathcal{A}') = \mathcal{A}'', \text{ alors } \mathcal{A}'' \subseteq \mathcal{A}'. \end{array} \tag{5}$$

L'objectif est de définir et d'implémenter une telle fonction, tel que le temps nécessaire au calcul de l'observation soit réduit. Le présent travail repose sur des implémentations existantes d'outils d'observation. Nous nous intéresserons donc à la complexité empirique moyenne, *i.e.*, le temps CPU moyen observé pour l'exécution de simulations implémentant les différentes méthodes d'observation. Cette complexité, notée $\mathcal{C}$, dépend du nombre d'agents, $|\mathcal{A}|$, et de l'espérance du cardinal du sous-ensemble $\mathcal{A}'$ de $\mathcal{A}$ calculé par $filtre$ (*i.e.*, dans notre exemple de N et de $E(Z)$). On cherche donc à déterminer des méthodes de filtrage telles que :

$$\mathcal{C}(obs'(filtre(\mathcal{A}))) < \mathcal{C}(obs(\mathcal{A})). \tag{6}$$

3.1 Auto-observation des agents

L'idée derrière cette méthode est d'implémenter la fonction de filtrage dans les agents de la simulation eux-mêmes. L'utilisation d'un modèle organisationnel permet alors d'identifier l'ensemble des agents devant être observés. Ainsi, un groupe est défini comme *l'ensemble des agents contenant les informations nécessaires et suffisantes au calcul d'une propriété*. En d'autres termes, un groupe définit pour une observation particulière, l'ensemble minimal des agents nécessaire à son calcul. Les agents s'auto-observent pour déterminer s'ils doivent rejoindre, quitter ou rester dans un groupe.

Un ensemble de règles (caractérisant $filtre$) spécifié par le modélisateur/observateur, déterminant si un agent doit être observé pour calculer la valeur d'une propriété particulière, est évalué à chaque pas de simulation. Ainsi, dans notre cas d'étude, on considère le groupe G, contenant l'ensemble des agents présents dans $\mathcal{Z}$ et on associe à chaque agent l'ensemble de règles suivant :

– si l'agent se trouve dans $\mathcal{Z}$ et que l'agent n'appartient pas à G, alors le rejoindre,
– si l'agent ne se trouve pas dans $\mathcal{Z}$ et que l'agent appartient à G, alors le quitter.

Le système d'observation (obs') ne sondera que les agents appartenant au groupe G. La figure 2 représente la différence entre les temps de calcul de simulations observées à l'aide des méthodes basée sur l'auto-observation des agents

et naïve en fonction du nombre d'agents dans la simulation N et du taux moyen d'agents observés $E(Z)/N$. La ligne en pointillée représente la ligne de niveau 0, *i.e.*, lorsqu'il n'y a pas de différence entre les deux méthodes d'observation. Ainsi, la zone située en dessous de cette ligne correspond aux cas où la méthode basée sur l'auto-observation des agents est la plus efficace. À l'inverse, la zone située au dessus de cette ligne correspond aux cas où la méthode naïve est la plus efficace.

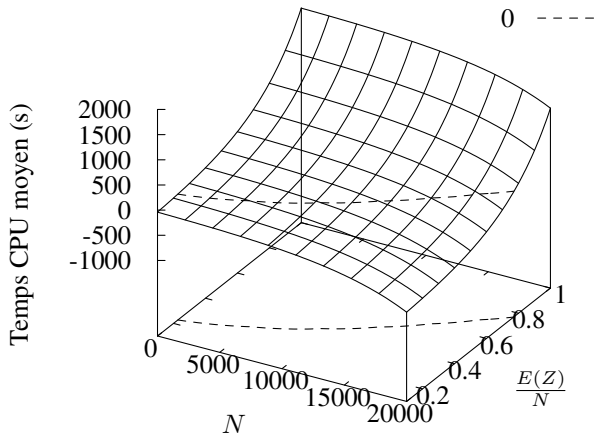

FIG. 2 – Différence entre les temps CPU moyens (s) de simulations observées à l'aide des méthodes basée sur l'auto-observation des agents et naïve

3.2 Échantillonnage de la population

En réécrivant l'équation 3 de la manière suivante :

$$obs(\mathcal{A}') \simeq obs(\mathcal{A}) \text{ et } \nexists\, \mathcal{A}'' \subset \mathcal{A}' \mid obs(\mathcal{A}'') \simeq obs(\mathcal{A}), \quad (7)$$

i.e., en autorisant des observations approximatives, il devient possible de filtrer la population d'agents à observer sur des critères statistiques. On ne considère donc plus de fonction d'observation spécifique, les agents retournés par la fonction de filtrage n'étant pas nécessairement les agents à observer.

La théorie des sondages[2] fournit un cadre formel permettant de déterminer la taille de l'échantillon à observer. Considérons le cas de l'exemple précédent et notons n le nombre d'agents dans l'échantillon d'agents observés.

[2]Les résultats de cette théorie exposés ici ne seront pas démontrés, le lecteur intéressé peut se référer à [2] pour une présentation exhaustive.

Un estimateur de Z, noté $\hat{Z}$, peut ainsi être construit à partir de cet échantillon. *E.g.*, l'estimateur de Horvitz-Thompson est défini par

$$\hat{Z} = \frac{N}{n} Z_n, \quad (8)$$

où Z_n représente le nombre d'agents de l'échantillon présents dans $\mathcal{Z}$. La qualité de cet estimateur, pour un échantillon donné, dépend de la variance V de $\hat{Z}$. Ici,

$$V(\hat{Z}) = (1 - \frac{n}{N}) \cdot \frac{S^2}{n} \simeq \frac{S^2}{n} \text{ si } N >> n. \quad (9)$$

S^2 peut être estimé par :

$$S^2 \simeq (1 - \frac{E(Z)}{N}) \cdot \frac{E(Z)}{N}. \quad (10)$$

La précision absolue de $\hat{Z}$ est estimée par la demi-longueur de l'intervalle de confiance à 95% : ϵ. On suppose que la loi de $E(Z)$ est une loi normale :

$$\epsilon \simeq 2 \cdot \sqrt{V(\hat{Z})}. \quad (11)$$

La taille de l'échantillon est déterminée à partir de l'erreur absolue maximale d tolérée pour l'observation. On cherche donc à minimiser n sous la contrainte $\epsilon \leq d$. Ainsi :

$$n = \frac{1}{\frac{d^2}{4S^2} + \frac{1}{N}}. \quad (12)$$

La figure 3 montre le nombre d'agents à observer pour une erreur absolue tolérée de 0.08 et pour $E(Z) = N/2$.

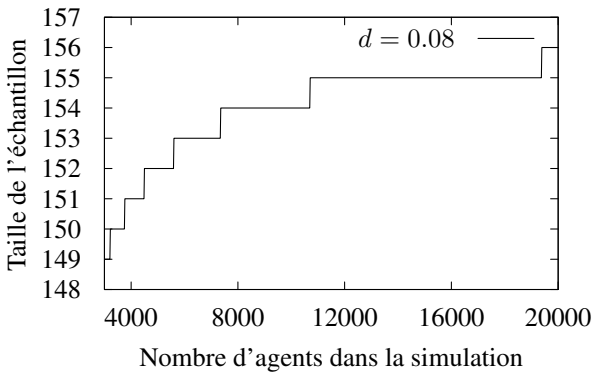

FIG. 3 – Taille de l'échantillon à observer pour $d = 0.08$

L'impact sur le temps de calcul est présenté dans la figure 4. La sémantique est la même que pour la figure 2 : la zone située à droite de la ligne en pointillés correspond aux cas où la méthode basée sur l'échantillonnage de la population d'agents est la plus efficace. À l'inverse, la zone située à gauche de cette ligne correspond aux cas où la méthode naïve est la plus efficace.

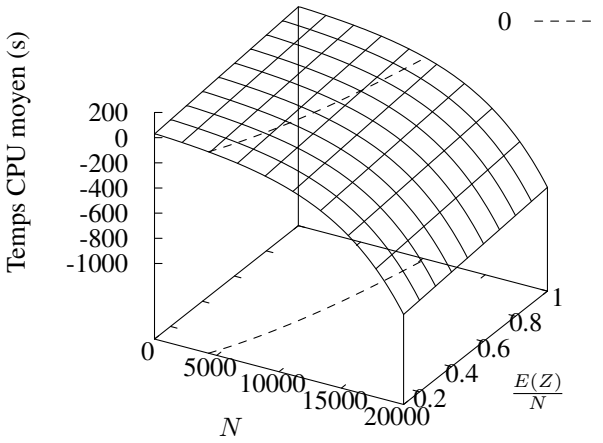

FIG. 4 – Différence entre les temps CPU moyens (s) de simulations observées à l'aide des méthodes basée sur l'échantillonnage de la population d'agents (avec $d = 0.08$) et naïve

3.3 Bilan

Pour conclure, la figure 5 synthétise les résultats précédents en identifiant les conditions dans lesquelles il est préférable, en termes de temps de calcul, d'utiliser une méthode d'observation particulière. Bien évidemment, ces résultats sont propres au cas d'étude considéré ; ils permettent cependant de mettre en évidence que le choix d'une méthode d'observation n'a rien de trivial et que les performances de ces méthodes doivent être analysées sur un ensemble de simulations préalablement à l'exploitation du modèle. De plus, il semble très probable que la forme générale de la figure 5 soit, indépendamment de l'échelle, invariante quel que soit le problème considéré.

De manière générale, lorsque la proportion d'agents à observer est faible ou lorsque le nombre d'agents dans la simulation est important, la méthode naïve n'est pas la plus efficace et il devient alors intéressant de mettre en œuvre des méthodes plus évoluées comme celles présentées dans cet article.

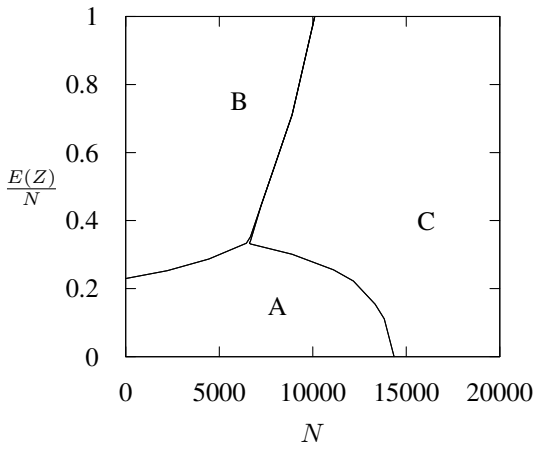

FIG. 5 – Conditions dans lesquelles les méthodes d'observations sont les plus efficaces ; A : auto-observation, B : naïve, C : échantillonnage

4 Conclusion et perspectives

Les méthodes de filtrage proposées dans cet article permettent, dans certaines conditions identifiées dans le cadre d'un problème jouet, de réduire les temps de calcul de SOA composées de très nombreux agents. Cependant, le filtrage ne semble pas être la seule stratégie d'optimisation. Ainsi, la nécessité, du point de vue de la qualité de l'observation, d'actualiser la valeur d'une propriété dépend de sa variation : grossièrement, plus elle est importante, plus la fréquence d'observation devrait l'être aussi. Or, les SOA sont utilisées pour simuler des systèmes complexes aux dynamiques souvent non linéaires. L'adaptation dynamique de la fréquence d'observation pourrait donc être une piste intéressante pour réduire le temps de calcul de SOA.

De plus, dans le cas de la méthode basée sur l'échantillonnage de la population (*cf.* section 3.2), seul un plan de sondage simple a été mis en place. Il semblerait intéressant de considérer des plans de sondages plus complexes (*e.g.*, un plan stratifié ou avec remise) afin d'observer des populations d'agents dans lesquelles la distribution des propriétés individuelles observées n'est pas uniforme. Un système d'apprentissage pourrait d'ailleurs être implémenté pour analyser l'impact des différents paramètres d'échantillonnage (taille de l'échantillon, fréquence d'échantillonnage, choix du plan de sondage adéquat) sur le temps de calcul et la qualité des observations. Similairement, le modèle organisationnel sur lequel re-

pose l'auto-observation est lui aussi simple : seule la notion de groupe a été utilisée. L'utilisation d'un modèle organisationnel plus complet (*e.g.*, AGR [3]) pourrait permettre de prendre en compte des observations plus complexes.

D'un point de vue méthodologique, la mise en place de l'une de ces méthodes d'observation peut aider à la définition de la validité d'une simulation ou du simulateur en comparant des valeurs de paramètres observables lors des simulations à celles de paramètres obtenus lors d'expérimentations par exemple. De tels travaux sont en cours d'étude pour la validation de simulations dans le cadre du développement d'un Système Informatique d'Aide à la Décision (SIAD) dédié à l'entomologie médico-légale [6]. Les gains en temps de calcul obtenus grâce à la mise en place de ces méthodes d'observation devraient ainsi permettre de prendre en compte une quantité plus importante de paramètres et augmenter la fiabilité des validations des simulations.

Références

[1] J.A. Botía, J.M. Hernansaez, and A.F. Gómez-Skarmeta. *Programming Multi-Agent Systems*, volume 4411 of *Lecture Notes in Computer Science*, chapter On the Application of Clustering Techniques to Support Debugging Large-Scale Multi-Agent Systems, pages 217–227. Springer Berlin Heidelberg, 2006.

[2] A-M. Dussaix and J-M. Grosbras. *Les sondages : Principes et méthodes*. Que sais-je ? Presses Universitaires de France, 1996.

[3] J. Ferber and O. Gutknecht. A meta-model for the analysis and design of organizations in multi-agent systems. In *Proceedings of the Third International Conference on Multi-Agent Systems (ICMAS98)*, pages 128–135, 1998.

[4] G. Kaminka, D. Pynadath, and M. Tambe. Monitoring teams by overhearing : A multi-agent planrecognition approach. *Journal of Artificial Intelligence*, 17 :83–135, 2002.

[5] F. Michel, G. Beurier, and J. Ferber. The turtlekit simulation platform : Application to complex systems. In *Proceedings of the First International Conference on Signal-Image Technology and Internet Based Systems*, pages 122–127, 2005.

[6] G. Morvan, D. Jolly, and D. Charabidze. Thermoregulation in p. terraenovae aggregations, an agent-based approach. In *Proceedings of the ESM'2008, Le Havre, France*, pages 417–422, October 27–29 2008.

[7] D. Payet, J.M. Medoc, T. Ralambondrainy, F., and R. Courdier. Outils d'observation et d'analyse de simulations multi-agents : l'expérience de la plate-forme geamas/biomas. In *CABM-HEMA. Conference on Multi-agent modelling for environmental management, Bourg Saint Maurice - Les Arcs, France*, Mars 2005.

[8] S. Railsback, S. Lytinen, and V. Grimm. Stupidmodel and extensions : A template and teaching tool for agent-based modeling platforms. *http ://condor.depaul.edu/ slytinen/abm/StupidModelFormulation.pdf*, 2005.

[9] T. Ralambondrainy, J.-M. Médoc, R. Courdier, and F. Guerrin. Tools to visualise the structure of multiagents' conversations at various levels of analysis. In *MODSIM 2007 International Congress on Modelling and Simulation. Modelling and Simulation Society of Australia and New Zealand*, pages 3098–3104, December 2007.

[10] R.E. Shannon. Introduction to the art and science of simulation. In *Proceedings of the* 30^{th} *conference on Winter simulation*, pages 7–14, 1998.

[11] D. Wilkins, T. Lee, and P. Berry. Interactive execution monitoring of agent teams. *Journal of Artificial Intelligence Research*, 18 :217–261, 2003.